package com.twitter.simclusters\_v2.score

import com.twitter.simclusters\_v2.score.WeightedSumAggregatedScoreStore.WeightedSumAggregatedScoreParameter

import com.twitter.simclusters\_v2.thriftscala.{

EmbeddingType,

GenericPairScoreId,

ModelVersion,

ScoreInternalId,

ScoringAlgorithm,

SimClustersEmbeddingId,

Score => ThriftScore,

ScoreId => ThriftScoreId,

SimClustersEmbeddingPairScoreId => ThriftSimClustersEmbeddingPairScoreId

}

import com.twitter.util.Future

/\*\*

\* A generic store wrapper to aggregate the scores of N underlying stores in a weighted fashion.

\*

\*/

case class WeightedSumAggregatedScoreStore(parameters: Seq[WeightedSumAggregatedScoreParameter])

extends AggregatedScoreStore {

override def get(k: ThriftScoreId): Future[Option[ThriftScore]] = {

val underlyingScores = parameters.map { parameter =>

scoreFacadeStore

.get(ThriftScoreId(parameter.scoreAlgorithm, parameter.idTransform(k.internalId)))

.map(\_.map(s => parameter.scoreTransform(s.score) \* parameter.weight))

}

Future.collect(underlyingScores).map { scores =>

if (scores.exists(\_.nonEmpty)) {

val newScore = scores.foldLeft(0.0) {

case (sum, maybeScore) =>

sum + maybeScore.getOrElse(0.0)

}

Some(ThriftScore(score = newScore))

} else {

// Return None if all of the underlying score is None.

None

}

}

}

}

object WeightedSumAggregatedScoreStore {

/\*\*

\* The parameter of WeightedSumAggregatedScoreStore. Create 0 to N parameters for a WeightedSum

\* AggregatedScore Store. Please evaluate the performance before productionization any new score.

\*

\* @param scoreAlgorithm the underlying score algorithm name

\* @param weight contribution to weighted sum of this sub-score

\* @param idTransform transform the source ScoreInternalId to underlying score InternalId.

\* @param scoreTransform function to apply to sub-score before adding to weighted sum

\*/

case class WeightedSumAggregatedScoreParameter(

scoreAlgorithm: ScoringAlgorithm,

weight: Double,

idTransform: ScoreInternalId => ScoreInternalId,

scoreTransform: Double => Double = identityScoreTransform)

val SameTypeScoreInternalIdTransform: ScoreInternalId => ScoreInternalId = { id => id }

val identityScoreTransform: Double => Double = { score => score }

// Convert Generic Internal Id to a SimClustersEmbeddingId

def genericPairScoreIdToSimClustersEmbeddingPairScoreId(

embeddingType1: EmbeddingType,

embeddingType2: EmbeddingType,

modelVersion: ModelVersion

): ScoreInternalId => ScoreInternalId = {

case id: ScoreInternalId.GenericPairScoreId =>

ScoreInternalId.SimClustersEmbeddingPairScoreId(

ThriftSimClustersEmbeddingPairScoreId(

SimClustersEmbeddingId(embeddingType1, modelVersion, id.genericPairScoreId.id1),

SimClustersEmbeddingId(embeddingType2, modelVersion, id.genericPairScoreId.id2)

))

}

val simClustersEmbeddingPairScoreIdToGenericPairScoreId: ScoreInternalId => ScoreInternalId = {

case ScoreInternalId.SimClustersEmbeddingPairScoreId(simClustersId) =>

ScoreInternalId.GenericPairScoreId(

GenericPairScoreId(simClustersId.id1.internalId, simClustersId.id2.internalId))

}

}